Profile related requests:

Get a user's profile information:

Request

Mimetype	application/json
Method	GET
URL	/api/profile/getProfile

Body Example	{
	username: "user"
	}

Response

Mimetype	application/json

Body Example: Individual	{
	"username": "user",
	"type": 0 ,
	"profile": {
	"name": "Bosco Njoku",
	"gender": "male",
	"birthdate": "1970-01-01",
	"phone": "100 1111 1010",
	"industry": "Software Engineering",
	"description": "I am a male homo-sapiens, currently working as a programmer. I'm learning to start my own online short video platform",
	"tags": ["software engineering", "programmer", "streaming"]
	}
	}

Body Example: Company	{
	"username": "user",
	"type": 1 ,
	"profile": {
	"name": "Bosco Network Co., Ltd.",
	"website": "bosco.io",
	"industry": "Entertainment",
	"description": "We are aiming to make short video entertainment more accessible than ever in Africa. ",
	"tags": ["entertainment", "short video", "streaming"]
	}
	}

Body Example: Partner	{
	"username": "user",
	"type": 2 ,
	"profile": {
	"name": "Adaora Din-Kariuki",
	"phone": "",
	"industry": "Software Engineering",
	"description": "I am looking for startup companies to invest in. ",
	"tags": ["software engineering", "investor"]
	}
	}

Remarks	● A field in "profile" will be *empty string* if it has yet not been filled by the user.

Get a user’s profile picture:

The backend will search under the /client/public/profilePics for the file <username>.png or <username>.jpeg, if neither exists then it will return a path to the default profile pic file.

Request

Mimetype	application/json
Method	GET
URL	/api/profile/getProfilePic
Params	username=user

Response

Mimetype	application/json

Body	{
	"url": "/profilePics/user"
	}

Remarks	• To display the profile pic in an <image> tag, set the *src* property of the <image> tag to the returned url.

Post a user’s changes in profile

The backend will get the user’s username from the token, then get the user type and update fields of the corresponding table. If a field is given in the request body but does not exist in the table of that user type, ignore it(It shouldn’t happen though).

Request

Mimetype	application/json
Method	POST
URL	/api/profile/updateProfile

Body Example	{
	"updates": {
	"name": "My New Name",
	"description": "My new description"
	}
	}

Remarks	● The "updates" may only include fields that the user changed. ● Validation will be done in the backend. ● This endpoint is not responsible for changing the user’s profile picture.

Response

Mimetype	application/json

Body Example	{
	"message": "success"
	}

Remarks	● On failure, the status will be 4xx and the message field of the response body will contain the corresponding error message.

Post a user's new profile picture

The backend will update the profile picture under /client/public/profilePics to the uploaded image.

Request

Mimetype	image/jpeg | image/png // Depends on the file format
Method	POST
URL	/api/profile/updateProfilePic
Body	<image data>
Remarks	• Make sure the file is not too big before sending the ajax request. • Make sure the user can only send .jpg/.jpeg and .png files. • If possible, make sure the dimension of the image is not too wild (Like, 60px*50px is ok but 810px*114514px is not). • Check this tutorial if you don't know how to send a request like this https://www.pluralsight.com/guides/using-react.js-and-jquery-to-update-a-profile-picture-with-a-preview This tutorial is using JQuery(the *$* function), but it is not necessary. Check the *fetch()* function and the ajax request using *fetch()* in the signup page.

Response

Mimetype	application/json

Body Example	{
	"message": "success"
	}

Community Related Requests

Get Post List

Requset

Method	GET
URL	/api/community/getPosts
URL Params	author?: string, author username, title?: string, title, post_before?: number, timestamp. Only posts made before this timestamp will be posted // Above are filter params post_per_page: number page_number: number

Response

Mimetype	application/json

Body Example	{
	post_count: 10
	posts: [
	{
	"author": "<username>",
	"post_time": 10230123801298,
	"id": "ajsdnfsvjbzxkv",
	"title": "My Title",
	"abbriv": "In this post, I …",
	"comment_count": 23
	},
	...
	]
	}

Get Post Content

Request

Mimetype	application/json
Method	GET
URL	api/community/getPostContent
Params	post_id: "aksdfbasdfasdf"

Response

Mimetype	application/json
Body Example	{ “id”: “Pxxxxxx”, "author": "<username>", "post_time": 10230123801298, "title": "My Title", "content": "This is the full post content.", "comments": [{ "id": "Casdaasdfbdkfa", "author": "username", "post_time": 168273691, "content": "This is my reply", "comments": [{ "reply_to": "reply_id", (optional, only used if this is a reply to a comment to a comment) "author": "username", "post_time": 198274923, "content": "This is my reply to reply" }, ...] }, ...] }

Make Post

Request

Mimetype	application/json
Method	POST
URL	/api/community/createPost

Body Example	{
	"title": "My post title",
	"body": "My post body. "
	}

Make Comment

Request

Mimetype	application/json
Method	POST
URL	/api/community/createComment

Body Example	{
	"reply_to": "Pasdfksdf" | "Csbkskdgffg",
	"body": "This is my reply"
	}

Delete Content

Request

Mimetype	application/json
Method	POST
URL	/api/community/deleteContent

Body Example	{
	"id": "Pasdfksdf" | "Csbkskdgffg",
	}

Follow Post

Request

Mimetype	application/json
Method	POST
URL	/api/community/followPost

Body Example	{
	"id": "Psbkskdgffg",
	"follow": true | false // false: unfollow
	}

E Learning

Create a new course

Method	PUT
URL	/api/learning/createCourse

Body Example	{
	“name”: “New Course”,
	“instructor”: “username”,
	“description”: “description of the new course”
	}

Remarks	In case of a name conflict, status 409 will be returned.

Edit a course

Method	POST
URL	/api/learning/updateCourse

Body Example	{
	“name”: “New Course”,
	“instructor”: “username”,
	“description”: “description of the new course”
	}

Delete a course

Method	DELETE
URL	/api/learning/deleteCourse

Body Example	{
	“name”: “New Course”
	}

Create a new module

Method	PUT
URL	/api/learning/createModule

Body Example	{
	“courseName”: “Course”,
	“moduleName”: “Module”
	}

Edit a module

Method	POST
URL	/api/learning/editModule

Body Example	{
	“moduleId”: “moduleid”,
	“newModuleName”: “Module New Name”
	}

Delete a module

Method	DELETE
URL	/api/learning/deleteModule

Body Example	{
	“moduleId”: “moduleid”
	}

Remarks	This will delete all contents in the module as well.

Create a video instance

Method	PUT
URL	/api/learning/createVideo

Body Example	{
	“name”: “My Video”,
	“description”: “description of the new video”,
	“url”: “” //youtube only
	}

Edit a video instance

Method	POST
URL	/api/learning/editVideo

Body Example	{
	“id”: “asdasckvnzxbcfawef”,
	“description”: “description of the new video”,
	“url”: “” // youtube only
	}

Delete a video instance

Method	DELETE
URL	/api/learning/deleteVideo

Body Example	{
	“id”: “asdasckvnzxbcfawef”
	}

Create a reading instance

Method	PUT
URL	/api/learning/createReading

Body Example	{
	“name”: “New Reading”,
	“description”: “This reading is for reading. ”
	}
	AND a file upload

Edit a reading instance

Method	POST
URL	/api/learning/editReading

Body Example	{
	“id”: “asdfasdfasdfgwr”,
	“name”: “New Reading Name”,
	“description”: “description of the new reading”
	}

Delete a reading instance

Method	DELETE
URL	/api/learning/deleteReading

Body Example	{
	“id”: “asdfasdfasasdf”
	}

Create a deliverable

Method	PUT
URL	/api/learning/createDeliverable

Body Example	{
	“name”: “[A1] Assignment 1”,
	“description”: “description of the new course”,
	“due”: <timestamp>
	}

Edit a deliverable

Method	POST
URL	/api/learning/editDeliverable

Body Example	{
	“id”:
	“description”: “description of the new course”
	}

Delete a deliverable

Method	DELETE
URL	/api/learning/deleteReadings

Body Example	{
	“name”: “New Course”
	}

Enroll in a course

Method	POST
URL	/api/learning/enrollCourse

Body Example	{
	“name”: “Course Name”
	}

Unenroll a course

Method	POST
URL	/api/learning/dropCourse

Body Example	{
	“name”: “Course Name”
	}

Get all courses

Requset

Mimetype	application/json
Method	GET
URL	/api/community/getCourses

Response

Mimetype	application/json
Body Example	[{ "instructor": "<username>", “name”: “Course Name”, “description”: “course description”, “enroled”: true }, ...]

Get course content

Request

Mimetype	application/json?
Method	GET
URL	/api/learning/getCourseContent
Params	courseName: “Course Name”

Response

Mimetype	application/json

Body Example	{
	"name": "Course Name",
	"instructor": "username",
	"description": "course description",
	"modules": [
	{
	"name": "module name",
	"contents": [
	{
	"type": "reading",
	"name": "lec01 reading",
	"description": "Read this",
	"path": "/files/Course Name/reading.txt"
	},
	{
	"type": "video",
	"name": "lec01 video" ,
	"description": "Watch this",
	"url": "http://youtube.ca/bruh"
	},
	{
	"type": "deliverable",
	"name": "Assignment 1",
	"description": "Do this",
	"due": 12837129471
	},
	...
	]
	},
	...
	]
	}

Submissions

Submit a submission to a deliverable

Method	POST
URL	/api/deliverable/createSubmission

Body Example	{
	"deliverableId": "wedsfadsjlifhaffio1231209ujdp",
	"content": "Comments"
	}
	*includes media file

Send Feedback

Method	POST
URL	/api/deliverable/sendFeedback

Body Example	{
	"submissionId": "81hf9oiugapw9j4389",
	"comments": "good job",
	"grade": 5
	
	}

Get Feedback

Method	GET
URL	/api/deliverable/getFeedback

Body Example	{
	"courseName", "Intro to CS",
	"submissionId": "81hf9oiugapw9j4389"
	}

Response

Mimetype	application/json

Body Example	{
	"grade": 5,
	"comment": "Great job"
	}

Get all submissions of a deliverable

Method	GET
URL	/api/deliverable/getSubmissions

Body Example	{
	"deliverableId": "cfo78yenwa7ya3hrfwc"
	}

Response

Mimetype	application/json

Body Example	{
	"onTime":[
	{
	"submissionId": "asfasdaddwe21das",
	"username": "asdad",
	"content": "ignore number 3a",
	"media": "files/asad.pdf",
	"posted": 1241321,
	"grade": 5
	},
	{
	"submissionId": "asfasdaddwe21das",
	"username": "gfasg",
	"content": "no comment",
	"media": "files/masda.pdf",
	"posted": 1243211,
	"grade": 6
	}
	]
	"late": [
	{
	"submissionId": "asfasdaddwe21das",
	"username": "ashfh",
	"content": "no comment",
	"media": "files/gd.pdf",
	"posted": 1249211,
	"grade": 2

	},
	{
	"submissionId": "asfasdaddwe21das",
	"username": "hafrr",
	"content": "no comment",
	"media": "files/GDSA.pdf",
	"posted": 1249211,
	"grade": 3
	}
	]
	}

Get the content of a submission

This is necessary because we cannot put deliverable files under public directory and let the browser directly access them, we need to check the identity of the requester.

Method	GET
URL	/api/learning/getSubmission

Body Example	{
	"submissionId": "81hf9oiugapw9j4389"
	}

Response

Mimetype	application/json

Body Example	{
	"username": "ad2wdw",
	"content": "Great job",
	"media": "files/${submissionId}.pdf",
	"posted": 123124,
	"grade": 5
	}

Calendar

Get all key dates related to a user

username -> courses -> key dates

Method	GET
URL	/api/calendar/getImportantDates

Body Example	{
	"courseName": "Intro to CS"
	}

Response

Mimetype	application/json

Body Example	{
	"dates": [12314512, 124451524]
	}